

Highload и Lowcode — единство и борьба противоположностей
Сергей Нуралиев, 1С
HL++
HighLoad++
2022



Вы можете использовать Lowcode-подход в ваших проектах!

* Мы говорим именно про подход, а не про готовые продукты / технологии





Зачем нужен Lowcode-подход?

- Повысить скорость создания приложений
- Снизить стоимость
- Понизить сложность владения
- Защита от проблем при смене команды разработчиков





Highload & Lowcode

Разные вселенные?

В своем докладе покажу:

- Это не так
- Когда это не так
- Зачем это не так





«Большой» Lowcode

Часто думают, что Lowcode – это для маленьких задач

Но существует Lowcode для крупных проектов и для облаков

Такого Lowcode в мире немного





Lowcode используют (пока?) не везде

В ERP и CRM такой подход встречается





Когда Lowcode-подход нужен?

- Много прикладной функциональности
- Много (десятки, сотни) разработчиков
- Хочется получить бонусы от Lowcode-подхода





Как совместить Lowcode и Highload?

У нас такой опыт есть – постараемся поделиться!





Немного о нас

Платформа **1С:Предприятие** – Lowcode-платформа для быстрой разработки различных бизнес-приложений

Она включает все (!), что нужно для этих задач





Клиенты хотели «больше»

Клиентам нравился наш подход и наше решение

Но они росли

-> нашим технологиям нужно было расти вместе с ними

Крупные клиенты тоже начали использовать наши приложения

-> и стали требовать от нас «соответствовать»



Рост нагрузки на платформу

Concurrent Users

Year	Concurrent Users
2003	80
2022	10000





Как мы дошли до такой жизни? ☺

Итог: постоянный рост требований по масштабируемости

Сейчас уже речь о самых крупных компаниях в РФ

-> нужна последовательная и постоянная борьба за Highload





Амбициозная цель

У нас:

- Достичь Highload, не отходя от Lowcode
- Lowcode нам терять нельзя – наша основная ценность

У вас может быть и наоборот:

- Получить Lowcode, не отходя от Highload





Может, разделить?

Нам часто предлагали: сделайте отдельную платформу для КОРП – мы не стали

Нам важно сохранять парадигму Lowcode – и в общем, и в деталях

Крайне нежелательно усложнять модель разработки

Платформа должна брать на себя максимум возможного





Идеал не всегда достижим

Не все задачи Highload можно решить без усложнения разработки

Иногда не получится скрыть природу вещей, т.к. они влияют на логику работы

Покажу на примерах

Но нужно стараться!



НЕМНОГО ПРИМЕРОВ





Load balancing

В бизнес-приложениях нагрузка очень неравномерная

Вызов от вызова по нагрузке может отличаться на порядки

По CPU, I/O, DBMS, дисковой подсистеме





Любимая задача на собеседованиях

«Спроектируйте балансировщик нагрузки»

Обычно рассказывают про Round-Robin, но вот в бизнес-приложениях он совсем плох ☹

А потом предлагают «пометить тяжелые запросы»

Но у нас Lowcode





Fault tolerance

Например, автоматический перезапуск рабочих процессов

Пользователи не должны ничего заметить!

Но это забота и о разработчиках!

Они плохо написали – а пользователи ничего не заметят!

Быстро поднятое не считается упавшим





Управляемые блокировки

Сначала использовали блокировки DBMS, отказались:

Нормально только на «небольших» / маленьких «средних» проектах

Иногда прямо безобразие – те же эскалации!

Потом сделали свое

А потом сделали свое заново☺





Наш механизм блокировок

Highload-сервис кластера

- Определение пересечений и помещение в ожидание
- Отслеживание deadlock'ов

Параллельность всегда, кроме случаев с требованиями по бизнес-логике

Как раз тот случай, когда нельзя скрывать природу вещей!

Но и тут Lowcode – управлять блокировками очень просто

Но можно и нужно думать головой о бизнес-логике ☺





Дата-акселератор (in-memory DB)

Codename: FimaDB

У заказчиков растут потребности в Аналитике

- Надо обрабатывать млн. записей за доли секунды
- «Живой анализ» не терпит задержек и ожиданий

Сделали свою in-memory DB

- «Очень реляционная»

Выдерживаем Lowcode (даже «ZeroCode»):

- Не требуется никаких усилий разработчиков – она просто работает
- Полная совместимость с другими DBMS





Механизм копий баз данных

Автоматическое распараллеливание читающей нагрузки

По сути – горизонтальное масштабирование DBMS

Борьба за скорость заливки и синхронизации данных

Тоже Lowcode – от разработчика ничего не требуется



Клиент

тонкий клиент

web-клиент

web-сервер

Мобильные клиенты

Сервер

Кластер серверов 1С

Отчеты

Отчеты

Отчеты

СУБД

СУБД

Дата-акселератор

Дата-акселератор

Дата-акселератор

Механизм копий БД





А еще…

- Хранилище бинарных данных
- Хранилище сеансовых данных
- Механизм ограничений ресурсов
- Миграция данных
- Решение систем линейных уравнений
- Кэширование
- Журнал регистрации со специальными индексами
- **Фоновые задания, регламентные задания**
- Многопоточность – **чтобы не было опасно и сложно**!





Приемы и опыт Lowcode для Highload





Высокоуровневая модель

Слой абстракции

- Работы с данными
- С пользовательским интерфейсом
- Правами

Позволяет проводить оптимизации, не меняя приложение





Высокоуровневая модель

Lowcode – это про уровень абстракции

Помогает реализовать DDD

Обеспечивает семантику предметной области и декларативный подход при разработке бизнес-приложений





Делаем Lowcode-слой

Плохая новость: часть приемов оптимизации становится недоступна

Хорошая новость: вы можете оптимизировать движок Lowcode-слоя

Сами приложения менять не потребуется





Не «ковыряйте дырочек»

Lowcode-слой храните максимально целостным

И не ведитесь на уговоры ☺

Иначе случится плохое…





Highload тоже бывает разный

В бизнес-приложениях бывает:

- *КОРП*
- *Облачный*

У них разные профили нагрузки

Это нужно учитывать и в концепции Lowcode





Low — не только code

Помимо простой разработки, нужно и простое администрирование

Красивого названия не придумали, мы иногда называем «zero-администрирование»

В примерах выше речь шла не только о разработчиках, но и об администраторах!





Безопасность!

Закладывайте в модель вопросы безопасности

И это именно про Lowcode

Это «защита» и от разработчика, и от пользователя





Не экономьте на инструментах

Мониторинг, логирование, диагностика и т.д.



Не упускайте бонус!

Доработали под капотом Lowcode-платформу —

все приложения на ней ощутили результат

Это дорогого стоит!





Что ещё нам предстоит довести до ума

- Очередь заданий со сложной балансировкой
- Скорость работы встроенного языка
- Row Level Security





Цифры

Строк в Платформе 1С

- более 16 млн

Строк в 1С:ERP

- более 12 млн
- если бы без Lowcode,
то 30 – 50 млн

Разработчиков на платформе 1С

- более 300 000

Количество пользователей

- до 150 000 АРМ
- до 10 000 конкурентных

Объёмы данных

- десятки ТБ
(знаем проект с 36 ТБ)



Примеры из жизни…



Почта России. 150 000 рабочих мест. 86 филиалов и 1000 структурных подразделений.



Трансмашхолдинг. 40 000 автоматизированных рабочих мест. ERP, MES, CPM, WMS, ECM, HRM, ITIL. Есть предприятия с более 5000 зарегистрированными пользователями ERP.



Государственная корпорация «Ростех». 30 000 пользователей. Централизованная система управления финансами и закупками 700 организаций. Ежедневно онлайн 7000 пользователей. 36 ТБ данных.

HL HighLoad++ 2022

Обратная связь
и комментарии по
докладу по ссылке



Спасибо!

